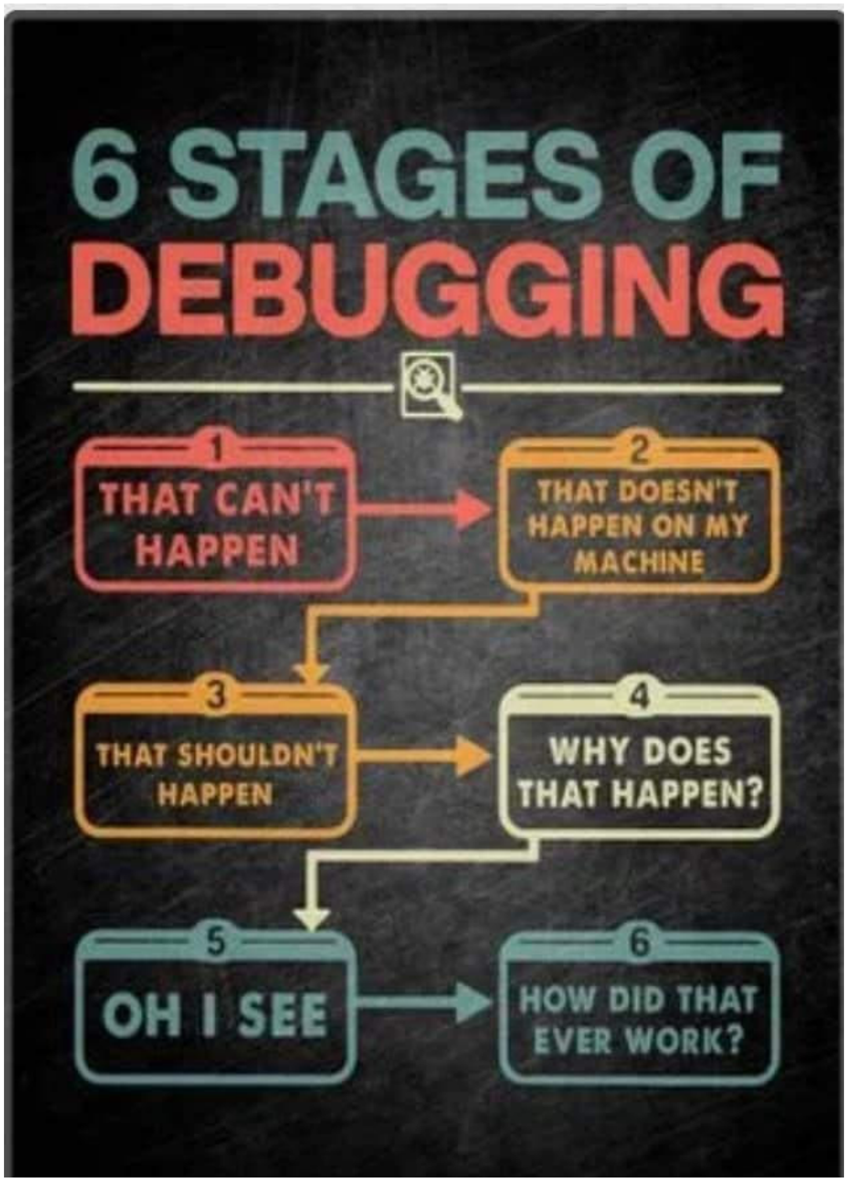
Security Now! #863 - 03-22-22
Use After Free

This week on Security Now!

This week we look at the US's new cybercrime reporting law that was just passed. We examine a worrisome software supply chain sabotage and the trend it represents. We look at "Browser-in-the-browser," a new way to spoof sign-in dialogs to capture authentication credentials, and we examine the way MicroTik routers are being used by the TrickBot botnet to obscure their command and control servers. A very concerning infinite loop bug has been uncovered in OpenSSL (time to update!) and CISA walks us through their forensic analysis of a Russian attack on an NGO. We then take a look at the Windows vulnerability that refuses to be resolved, and we'll finish by spending a bit more time than we have so far looking more closely at why User-After-Free flaws continue to be so challenging.



Security News

Report Cybercrime: It's the Law.

Last Tuesday, a $1.5 trillion government funding bill was signed into law by President Biden. We're talking about that here because packaged into the legislation is something we've touched on a few times when it has periodically been discussed. But what was under consideration until now is now law. That new law mandates that owners and operators of critical U.S. infrastructure must report when their organization has been hacked, and if a ransomware payment has been made.

This so-called "Strengthening American Cybersecurity Act", which was attached to the spending bill, requires that critical infrastructure operators alert the Homeland Security Department's CISA (our Cybersecurity and Infrastructure Security Agency) within 3 days (72 hours) of a breach, and within 24 hours if the organization made a ransomware payment. This new legislation also grants CISA subpoena power, allowing it to compel testimony from any entities which do not report a cyber incident or ransomware payment.

Interestingly, this move is a reversal in recent policy because it was removed from (or "stripped out of" as legislators like to say) the U.S.'s recent annual defense policy bill which was signed a few months ago. One wonders whether the potential for attacks against U.S. infrastructure, secondary to the Ukraine/Russian conflict, might be behind this change of heart? And my eyebrows rose when I learned that this "Strengthening American Cybersecurity Act" passed in the Senate with a unanimous vote.

So now, CISA will have up to two years to publish a notice in the Federal Register on proposed rulemaking to implement the reporting effort. However, reporting on this has suggested that CISA might choose to move faster thanks to the current heightened threat climate.

The Senate's Homeland Security Committee Chair Gary Peters, who authored and championed the legislation along with Sen. Rob Portman, issued a statement saying: "This historic, new law will make major updates to our cybersecurity policy to ensure that, for the first time ever, every single critical infrastructure owner and operator in America is reporting cyber-attacks and ransomware payments to the federal government."

Additionally, Rob Portman, the panel's top Republican, said the legislation will "give the National Cyber Director, CISA, and other appropriate agencies broad visibility into the cyberattacks taking place across our nation on a daily basis to enable a whole-of-government response, mitigation, and warning to critical infrastructure and others of ongoing and imminent attacks."

As I was researching this I got to wondering what exactly constituted critical U.S. infrastructure? Who exactly would be required to report? We all hear the term and phrase "critical infrastructure" bandied about all the time, but what's in there? To answer that question, I attempted to read some of the actual legislation — I don't recommend it — because now I have even less idea how our government actually works and more amazement that for the most part it seems to. I did manage to track down Presidential Policy Directive 21, Section 2242, subsection 'b'. And under the heading "Designated Critical Infrastructure Sectors and Sector-Specific Agencies", it explains:

This directive identifies 16 critical infrastructure sectors and designates associated Federal SSAs (Sector-Specific Agencies). In some cases co-SSAs are designated where those departments share the roles and responsibilities of the SSA. The Secretary of Homeland Security shall periodically evaluate the need for and approve changes to critical infrastructure sectors and shall consult with the Assistant to the President for Homeland Security and Counterterrorism before changing a critical infrastructure sector or a designated SSA for that sector. The sectors and SSAs are as follows (in alphabetical order):

Chemical, Commercial Facilities, Communications, Critical Manufacturing, Dams, Defense Industrial Base, Emergency Services, Energy, Financial Services, Food and Agriculture, Government Facilities, Healthcare and Public Health, Information Technology, Nuclear Reactors, Materials, and Waste, Transportation Systems, Water and Wastewater Systems.

So. Bottom line is reporting attacks and extortion payments is no longer optional. It's the law.

A software supply chain compromise

Winning this past week's prize for the most listener-reported incident was a very worrisome software supply-chain event. I don't think it qualifies for being called an attack since it was an inside job. Consequently most of the tech is using the term “sabotage.”

NPM is the package manager for today's #1 most popular, most used and most in demand programming language: JavaScript. NPM is the default package manager for JavaScript's super-popular runtime environment Node.js. And a very prominent Node module known as Node-IPC is commonly used for local and remote inter-process communication (IPC) with support for Linux, macOS, and Windows. Node-PCI has over 1.1 million weekly downloads.

In the context of Java's Log4j incident and vulnerability, we were talking about code dependency trees. The idea being that one software library relies upon several others and they may rely upon others, etc.

Liran Tal at “snyk” was the first to report their discovery of the problem. Liran wrote that:

On March 15, 2022, users of the popular Vue.js frontend JavaScript framework started experiencing what can only be described as a supply chain attack impacting the npm ecosystem. This was the result of the nested dependencies node-ipc and peacenotwar being sabotaged as an act of protest by the maintainer of the node-ipc package.

This security incident involves destructive acts of corrupting files on disk by one maintainer and their attempts to hide and restate that deliberate sabotage in different forms. While this is an attack with protest-driven motivations, it highlights a larger issue facing the software supply chain: [that] the transitive dependencies in your code can have a huge impact on your security.

The story started on March 8, 2022, at which time npm maintainer “RIAEvangelist”, whose real-world name is Brandon Nozaki Miller, wrote source code and published an npm package named “peacenotwar”, which describes itself:

This code serves as a non-destructive example of why controlling your node modules is important. It also serves as a non-violent protest against Russia's aggression that threatens the world right now. This module will add a message of peace on your users' desktops, and it will only do it if it does not already exist just to be polite.

It’s interesting that Brandon writes that he’ll only place his anti-war message on desktops that have not already received the message <quote> “just to be polite” because what Brandon then does is anything but polite...

His “peacenotwar” package sits around for a week with hardly any downloads. Then he adds this “peacenotwar” module as a dependency to one of his other very popular modules... that’s right, Node-IPC.

Then, apparently still getting even more amped up over the Russia/Ukraine conflict and feeling that he has the power to become more proactive, Brandon then added some deliberately base-64 obfuscated sabotage code to his Node-IPC package. This updated release added a function to check the IP address of developers who were using Node-IPC in their own projects. When an IP address geolocated to either Russia or Belarus, the new version wiped file contents from the machine, replacing them with a heart emoji.

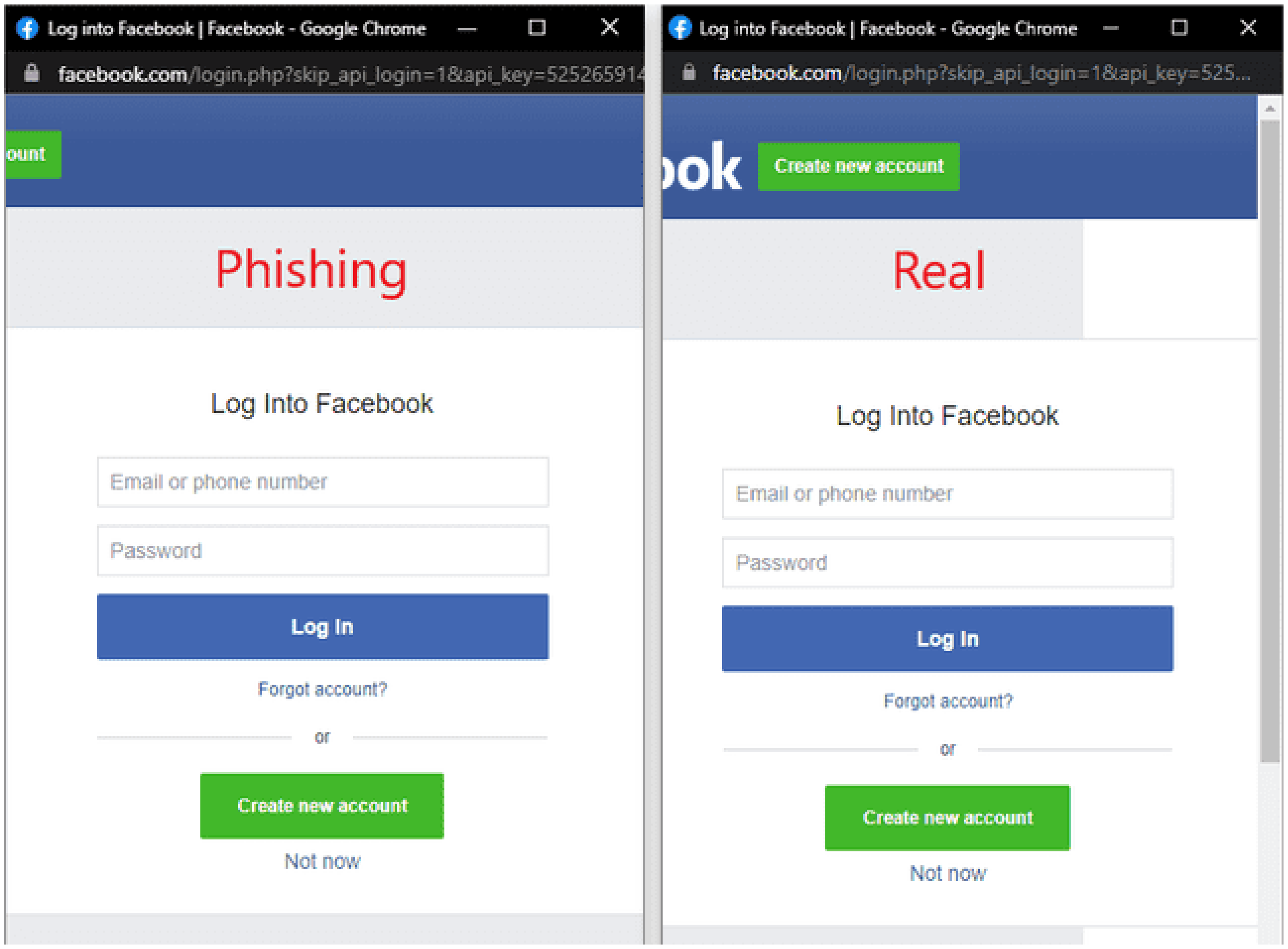
There’s obviously quite a lot that’s wrong with this. But on top of it just being plain wrong to take advantage of one’s implicitly trusted position as a package maintainer to have one’s package do something that it was clearly not sanctioned to do, Brandon’s targets were being very poorly chosen. If we are to believe what we’re told about the war sentiments of the Russian population, it’s the younger coders and developers who are more likely to be using Russian propaganda bypass measures, and to have a more Western view of what’s actually happening. And along comes some idiot who wipes out a sympathetic developer’s files only because of where they are coding.

But, sad as all that is, even more sadly, Brandon is not alone. The count of similarly recently altered software has risen to 21 separately identified pieces. I noted that the popular GUI toolkit QT is on the list, and another is es5-ext, which provides code for the ECMAScript 6 scripting language specification. A new dependency named postinstall.js, which the developer added on March 7, checks to see if the user’s computer has a Russian IP address, in which case the code broadcasts a “call for peace.” So while not all of these changes are destructive, they are almost certainly not what those who are using them want.

Brandon quickly removed his malicious changes due after a very predictable outcry arose from other enraged developers. And he certainly has not enhanced his credibility or reputation.

Browser in the Browser:
As I start to describe this latest attack I have to conclude that the bad guys really do have an advantage. They're able to sit back and steeple their fingers with a faraway look in their eyes, while they endlessly cook up one sneaky way after another to bend the technology we’ve innocently given them to their dastardly purposes.

Today, we have the latest attack concept that's been named “Browser-in-the-browser.” It’s a novel new form of phishing attack designed and documented by a pen tester who goes by the handle “mr.d0x”. It’s obvious once you see it, but it would not be obvious to a user who encounters it. It’s able to perfectly and convincingly simulate the pop-up browser window that we’re all encountering more and more often with the spread of those third-party single sign-on (SSO) options embedded on websites such as "Sign in with Google" (or Facebook, Apple, or Microsoft).



As the two screenshots above demonstrate, today’s web page rendering allows JavaScript, CSS & HTML to perfectly recreate the appearance of a legitimate “Sign in with....” pop-up dialog.

mr.d0x notes that one of the things that makes this attack so effective is that the spoofed dialog clearly and prominently displays the exactly correct domain name, the verification of which we’ve been drilling into our users for years. *“Make sure you’re at the right site. Double-check that URL for typos or look-alike characters. Don’t be fooled.”* And then along comes this perfect replica of a third-party sign-in dialog that accepts their innocently entered username and password.

And, to avoid any unnecessary duplication of effort, mr.d0x has thoughtfully provided attacker-ready easy-to-use downloadable templates on Github for Chrome on either Windows or MacOS and in either dark mode or light mode: https://github.com/mrd0x/BITB

TrickBot, MicroTik & Microsoft

Last Wednesday, Microsoft blogged under the title: *"Uncovering Trickbot's use of IoT devices in command-and-control infrastructure."* Microsoft considers MicroTik consumer routers to be IoT devices. I suppose that since they lack any keyboard and display and they are by definition connected to the Internet, they qualify. So Microsoft offered an interesting forensic write up which I'll share the best bit of. They begin with a bit of history:

Trickbot, a sophisticated trojan that has evolved significantly since its discovery in 2016, has continually expanded its capabilities and, even with disruption efforts and news of its infrastructure going offline, it has managed to remain one of the most persistent threats in recent years. The malware's modular nature has allowed it to be increasingly adaptable to different networks, environments, and devices. In addition, it has grown to include numerous plug-ins, access-as-a-service backdoors for other malware like Ryuk ransomware, and mining capabilities. A significant part of its evolution also includes making its attacks and infrastructure more durable against detection, including continuously improving its persistence capabilities, evading researchers and reverse engineering, and finding new ways to maintain the stability of its command-and-control (C2) framework.

This continuous evolution has seen Trickbot expand its reach from computers to Internet of Things (IoT) devices such as routers, with the malware updating its C2 infrastructure to utilize MikroTik devices and modules. MikroTik routers are widely used around the world across different industries. By using MikroTik routers as proxy servers for its C2 servers and redirecting the traffic through non-standard ports, Trickbot adds another persistence layer that helps malicious IPs evade detection by standard security systems.

When Microsoft says that MicroTik routers are being used as command and control proxies using non-standard ports they mean that once they have managed to crawl inside a MicroTik router — we'll discuss exactly how that happens in a minute — they setup a proxy server that's listening on one or more non-standard ports for incoming C2 queries from instances of TrickBot in the field. This provides the attackers with much more isolation and resilience than if all instances of TrickBot are phoning home directly. We've all seen the movies where tracing the traffic is thwarted by having it bouncing around among various intermediaries in widely spread jurisdictions. And also, assuming that there's a sufficient available inventory of vulnerable MicroTik devices, having different instances of deployed Trickbots connecting to widely dispersed MicroTik proxies makes finding the actual command and control servers much more difficult and it makes the Trickbot botnet much more difficult to take down.

The Microsoft Defender for IoT research team has recently discovered the exact method through which MikroTik devices are used in Trickbot's C2 infrastructure. In this blog, we will share our analysis of the method and provide insights on how attackers gain access to MikroTik devices and use compromised IoT devices in Trickbot attacks. This analysis has enabled us to develop a forensic tool to identify Trickbot-related compromise and other suspicious indicators on MikroTik devices. We published this tool to help customers ensure these IoT devices are not susceptible to these attacks. We're also sharing recommended steps for detection and remediating compromise if found, as well as general prevention steps to protect against future attacks.

Okay. There are three ways Microsoft has detected attackers gaining an initial foothold on MicroTik devices:

They will first attempt to gain access using default MikroTik passwords. It turns out that the old lesson has still not been learned, and that MicroTik's design wrongly provides for remote access with default credentials. As we know, default credentials, if any must exist, should never also be allowed to be used to enable remote authentication from the WAN interface. Any user wishing to enable WAN-side remote admin, after first being clearly and carefully warned that it's a "Really Bad Idea", should be required to create non-default credentials. Not doing this is incredibly poor security design in this day and age.

So, Microsoft will first try using the device's default credentials. If that fails then a brute force "credential stuffing" attack is attempted. Microsoft wrote that they have seen attackers using unique passwords that were probably harvested from other MikroTik devices. Hey, it's worth a shot.

And, believe it or not (oh yeah, I know, we all believe it) by exploiting CVE-2018-14847, a well known and long since patched 4 year old vulnerability which will succeed on devices still running RouterOS versions older than v6.42, there's another way in. This vulnerability, which we covered back when it was news at the time, gives an attacker the ability to read arbitrary files from a remote MicroTik router. Specifically, a file such as **user.dat**, which contains passwords.

And once in, they then change the router's password to retain control. Since any such router is probably long forgotten, tucked away in some dark corner or a closet with a sign on the door which reads: *"Don't mess with anything in here"* it's unlikely that anyone will be inconvenienced by having their router's password unilaterally changed, let alone having it commandeered for use as a botnet command and control proxy.

The attackers then issue a handy dandy MicroTik command that redirects traffic between two external ports on the router, thus enabling and establishing a line of communication between Trickbot-infected systems and its command and control — or perhaps another indirect link in the command and control chain if they want to be more fancy like in the movies. MicroTik has a RouterOS containing very powerful SSH-accessible IP, firewall and NAT stacks. It's this unique remotely accessible static NAT capability that makes MicroTik routers so useful as unattended communication proxies.

Microsoft says that they monitored a Trickbot-infected device connecting to a MikroTik router over port 449, after which the MicroTik router connected to a Trickbot command and control server over port 80.

And as for today's inventory of MikroTik routers, a relatively recent report by Eclypsium published this last December, found that **hundreds of thousands** of MikroTik routers are **still vulnerable** to remote exploitation, despite patches having been available for years. And because these devices feature unusually powerful hardware, they're seen as high-value targets.

Back in early December of last year, 2021, when covering this report from Eclypsium, our friends over at BleepingComputer reached out to MicroTik for some comment about the sad state of their otherwise very nice and quite powerful router's updating. MicroTik replied:

The Eclypsium report deals with the same old vulnerabilities we have mentioned in our previous security blogs. As far as we know - there are no new vulnerabilities in RouterOS. Furthermore, RouterOS has been recently independently audited by several third parties. They all arrived at the same conclusion.

Unfortunately, publishing updated firmware does not immediately protect the affected routers. We don't have an illegal backdoor to change the user's password and check their firewall or configuration. These steps must be taken by the users themselves.

We try our best to reach out to all users of RouterOS and remind them to do software upgrades, use secure passwords, check their firewall to restrict remote access to unfamiliar parties, and look for unusual scripts. Unfortunately, many users have never been in contact with MikroTik and are not actively monitoring their devices. We cooperate with various institutions worldwide to look for other solutions as well.

Meanwhile, we want to stress the importance of keeping your RouterOS installation up to date once again. That is the essential step to avoid all kinds of vulnerabilities.

I suppose we might be more sympathetic if they didn't also, apparently, have a default password that allows users to login remotely. Perhaps that's also been changed in their later releases. But again, so what? What may have started off as fiction in the movies, is actually underway today.

The Infinite Loop OpenSSL Bug

A new and moderately severe problem has been discovered in all supported versions of OpenSSL. It's officially being called a high severity problem, but since problems in OpenSSL can be so much worse than an infinite loop, I think it's more like moderately severe. Nevertheless, especially in today's current climate of increased cyber hostilities, it does seem destined to be exploited because it is so powerful, so ubiquitous and so easy to exploit. The reason is that any Internet server or service running any Internet protocol whose security is being provided by OpenSSL, as most UNIX- and Linux- and a great many cloud-based services are, that has not just received update patches, as many Internet servers will not have, which consumes a user-provided certificate, such as a client certificate, can be taken down with a simple deliberately-crafted TLS handshake.

The trouble was discovered and reported by Google's Tavis Ormandy who reported his findings to the OpenSSL team on the 24th of February, 2022. The trouble Tavis found and worked out with David Benjamin, Chrome's TLS guy, is that a modular square root function named "BN_mod_sqrt()", if provided with a maliciously crafted certificate to parse, will enter an infinite loop — thus permanently hanging the thread. The CVE write up is interesting. It explains...

*The BN_mod_sqrt() function, which computes a modular square root, contains a bug that can cause it to loop forever for non-prime moduli. Internally this function is used when parsing certificates that contain elliptic curve public keys in compressed form or explicit elliptic curve parameters with a base point encoded in compressed form. It is possible to trigger the infinite loop by crafting a certificate that has invalid explicit curve parameters. Since certificate parsing happens prior to verification of the certificate signature, **any process that parses an externally supplied certificate may thus be subject to a denial of service attack**.*

In other words, since the certificate's signature is checked for validity after the exploitable parameters are processed, anyone can trivially create a certificate that will completely lock-up anything that's using OpenSSL. Their description continues:

The infinite loop can also be reached when parsing crafted private keys as they can contain explicit elliptic curve parameters. Thus vulnerable situations include:

- *TLS clients consuming server certificates.*
- *TLS servers consuming client certificates.*
- *Hosting providers taking certificates or private keys from customers.*
- *Certificate authorities parsing certification requests from subscribers.*
- *Anything else which parses ASN.1 elliptic curve parameters.*

And any other applications that use the BN_mod_sqrt() where the attacker can control the parameter values are vulnerable to this DoS issue.

In the [earliest] *OpenSSL 1.0.2 version, the public key is not parsed during initial parsing of the certificate which makes it slightly harder to trigger the infinite loop. However any operation which requires the public key from the certificate will trigger the infinite loop. In particular the attacker can use a self-signed certificate to trigger the loop during verification of the certificate signature.*

This issue affects OpenSSL versions 1.0.2, 1.1.1 and 3.0. It was addressed in the releases of 1.1.1n and 3.0.2 on the 15th March 2022 [Last Tuesday a week ago]. *Fixed in OpenSSL 3.0.2 (Affected 3.0.0,3.0.1). Fixed in OpenSSL 1.1.1n (Affected 1.1.1-1.1.1m).*

Note that OpenSSL 1.0.2, which **is** vulnerable, has reached EOL and is not actively supported, so users are advised to upgrade to a new release branch as soon as possible. It will not be fixed.

Note, also, that many critical infrastructure'ish things routinely make use of certificate-based mutual authentication, where each of the two endpoints provides an identity-asserting and assuring certificate as a means for securely establishing their identity. Therefore, any such system with listening ports exposed to the public Internet, despite its administrators being sure that it will only connect to other endpoints which provide the proper matching certificate, are immediately vulnerable to being taken down by anyone else on the Internet.

Not surprisingly, this has generated a great deal of interest, and Github has been buzzing. All the work has been done, with someone using the handle "catbro666" providing a working proof of concept certificate which will take down any unpatched instance of OpenSSL that makes the mistake of accepting any such certificate. The hosting processor is pinned at 100% utilization and nothing else gets done.

So far, there are no confirmed indications of exploitation, but for this one its exploitation sooner than later seems way more likely than not.

CISA Alert AA22-074A
I want to share an interesting CISA alert which details a Russian state-supported attack upon an NGO (a non-governmental organization) because it contains some interesting and quite specific forensic investigation detail and may have some useful takeaways for some of our listeners.

CISA's alert is titled: *"Russian State-Sponsored Cyber Actors Gain Network Access by Exploiting Default Multifactor Authentication Protocols and "PrintNightmare" Vulnerability"*

CISA's report opens with a summary:

The Federal Bureau of Investigation (FBI) and Cybersecurity and Infrastructure Security Agency (CISA) are releasing this joint Cybersecurity Advisory (CSA) to warn organizations that Russian state-sponsored cyber actors have gained network access through exploitation of default MFA protocols and a known vulnerability. As early as May 2021, Russian state-sponsored cyber actors took advantage of a misconfigured account, set to default MFA protocols at a non-governmental organization (NGO), allowing them to enroll a new device for MFA and access the victim network. The actors then exploited a critical Windows Print Spooler vulnerability, "PrintNightmare" (CVE-2021-34527) to run arbitrary code with system privileges. Russian state-sponsored cyber actors successfully exploited the vulnerability while targeting an NGO using Cisco's Duo MFA, enabling access to cloud and email accounts for document exfiltration.

This advisory provides observed tactics, techniques, and procedures, indicators of compromise (IOCs), and recommendations to protect against Russian state-sponsored malicious cyber activity. FBI and CISA urge all organizations to apply the recommendations in the Mitigations section of this advisory.

Okay, So here are some interesting details of the attack which were uncovered during its investigation:

As early as May 2021, the FBI observed Russian state-sponsored cyber actors gain access to an NGO, exploit a flaw in default MFA protocols, and move laterally to the NGO's cloud environment.

Russian state-sponsored cyber actors gained initial access to the victim organization via compromised credentials and enrolling a new device in the organization's Duo MFA. The actors gained the credentials via brute-force password guessing attack, allowing them access to a victim account with a simple, predictable password. The victim account had been un-enrolled from Duo due to a long period of inactivity, but was not disabled in the Active Directory. As Duo's default configuration settings allow for the re-enrollment of a new device for dormant accounts, the actors were able to enroll a new device for this account, complete the authentication requirements, and obtain access to the victim network.

Using the compromised account, Russian state-sponsored cyber actors performed privilege escalation via exploitation of the "PrintNightmare" vulnerability to obtain administrator privileges. The actors also modified a domain controller file, *"c:\windows\system32\drivers\etc\hosts", thus redirecting Duo MFA calls to localhost instead of the Duo server. This change prevented the MFA service from contacting its server to validate MFA login—this effectively disabled MFA for active domain accounts because the default policy of Duo for Windows is to "Fail open" if the MFA server is unreachable.* [Then, as if to apologize for mentioning this, CISA added:] *Note: "fail open" can happen to any MFA implementation and is not exclusive to Duo.*

[But, since "the tyranny of the default" is one of this podcast's core observations, I was curious about Cisco's default for their DUO MFA. And, sure enough, in an FAQ over at duo.com, in answer to the question "How can I configure the fail mode?" Duo replies: *"By default, Duo Authentication for Windows Logon will "fail open" and permit the Windows logon to continue if it is unable to contact the Duo service. You can set the fail mode during installation to "fail closed" by deselecting the "Bypass Duo authentication when offline" box during installation."* But, come on, you're installing it for the first time and you don't know it well, yet. So you're going to leave all of the presumably recommended defaults alone, at least for the time being. We've all been there. Is there any reminder in the UI that that's the way it was set? No. Okay... but it's easy to change that later, right? No. Since they're answering the question "How can I configure the fail mode?" (Meaning that presumably it's not obvious, thus needing to be answered in the product's FAQ) Duo then proceeds to explain, "To change the fail mode after installation, use the Registry Editor (regedit.exe) with administrator privileges to create or update the following registry value... Wow. Not really the "Secure By Default" operation that you'd expect from an add-on MultiFactor Authentication system whose entire purpose is to meaningfully increase the system's security. After all, this system is now inconveniencing everyone all the time. Well, except for the Russians who simply bypassed it by taking strategic advantage of Duo's well-known default fail-open policy. If any of our listeners work with companies using Cisco's Duo MFA, it might be worth having your IT administration check out the current setting of that registry key. I've included a link to the FAQ question in the show notes: https://duo.com/docs/rdp-faq#how-can-i-configure-the-fail-mode? CISA continues...]

After effectively disabling MFA, Russian state-sponsored cyber actors were able to successfully authenticate to the victim's virtual private network (VPN) as non-administrator users and make Remote Desktop Protocol (RDP) connections to Windows domain controllers. The actors ran commands to obtain credentials for additional domain accounts; then using the method described in the previous paragraph, changed the MFA configuration file and bypassed MFA for these newly created and compromised accounts. The actors leveraged mostly internal Windows utilities already present within the victim network to perform this activity.

Using these compromised accounts without MFA enforced, Russian state-sponsored cyber actors were able to move laterally to the victim's cloud storage and email accounts and access desired content.

For those who aren't clear about the nature of NGOs — the target of this attack — they are typically benign, non-profit and explicitly unaffiliated with any particular government. A few examples of NGOs would be Greenpeace International, CARE, the World Wildlife Fund and Doctors without Borders. CISA doesn't tell us which NGO was targeted in this instance.

The Windows Local Privilege Escalation that Microsoft seems unable to fix
So this is the, believe it or not, ongoing saga of a Windows flaw that Microsoft, paradoxically, appears to be unable to fix. Since there's no obvious reason why this particular problem should be beyond them, one must conclude again, as we have been with increasing frequency lately, that they just don't really care, or that something has gone very wrong somewhere deep inside.

This all begins way back last August of 2021 with an elevation of privilege vulnerability in the Windows user profile service. Back then, it was tracked as CVE-2021-34484 after having been discovered by a security researcher named Abdelhamid Naceri. And if his name seems familiar it's because the problem he found just won't go away and we've been following along.

And it's not a small problem. It's a local privilege escalation which allows unprivileged users, like those pesky Ruskies who manage to gain access, to then acquire valuable administrative privileges in Windows 10, 11, and Server. It has a CVSS score of 7.8. And although exploits have been publicly disclosed, thus by Microsoft's own definition it's a 0-day, no evidence has yet been found of its exploitation in the wild — or any sincere attempt on Microsoft's part to actually fix it.

It was originally said to be fixed, as I noted, as part of August 2021's Patch Tuesday. But Naceri became curious after those patches were published and pushed out, to see how Microsoft had fixed what he had found. What he NOW found was that Microsoft had not fixed the problem. Instead, it was another one of those, "address the symptom not the problem" pseudo-fixes.

So, Naceri presented another proof of concept (PoC) which bypassed Microsoft's non-fix across all Windows desktop and server versions. And, spoiler alert... that remains true today.

It was at this point that our friends over at 0patch, the guys who create those wonderful itty-bitty micropatchs which fix problems that Microsoft hasn't yet, can't or won't, decided to release one of their own unofficial patches for all Windows versions, making it free for all of their registered users.

Apparently not being in any huge hurry to fix this after their August fail, the first time they checked this off as "fixed", Microsoft finally responded to this second bypass in their January 2022 Patch Tuesday. And they also updated its CVE to 2022-21919, again marking it as fixed. And again, believe it or not, it wasn't and still isn't.

Once again, Naceri found a way to bypass that second attempt and noted that this one was even worse than the first. Happily, that original micropatch still worked. But March's update two weeks ago changed the DLL in question ("profext.dll") and broke 0patch's micropatch. So not only did Microsoft's March update still not resolve the problem, but it also removed the 3rd-party protection that 0patch users had been enjoying until then.

Undaunted, 0patch has updated its micropatch so that its users are again protected while we presumably wait for Microsoft to yet again try to actually fix the problem.

In one happy bit of news, users of Windows 10 editions 1803, 1809 and 2004 have remained safely protected by 0patch's original patch, since those Windows editions have reached the end of their support life and were therefore spared from receiving Microsoft's update which re-exposed everyone else to this problem.

Use After Free

So, today's nominal podcast topic was inspired by several pieces of recent listener feedback, and it was a tweet from Stephen Bellchester that finally caused me to decide to give this a bit more time.

Closing the Loop

Stephen Bellchester @chesterm8
"It would be good if you could clarify your comments about use after free vulnerabilities in firefox (and similar ones in previous weeks). You have said a few times that use after free issues are being caused by memory management being done automatically by the language. That implied (to me at least) that the language that firefox is written in is memory managed and causing the issues, which seemed pretty unlikely to me. After some further digging it seems like the issue was actually with firefox's automatic memory management that runs underneath the HTML/CSS/JS stack."

Okay, so first of all, if I didn't say it, I would have meant that memory management is being performed by the language's runtime. JAVA is a perfect example, both because, being a virtual machine (JVM) based language, it has a clear runtime environment, and because it's famous for having a powerful garbage collector.

A web page is titled "How Java Garbage Collection Works" and it starts off:

Java Memory Management, with its built-in garbage collection, is one of the language's finest achievements. It allows developers to create new objects without worrying explicitly about memory allocation and deallocation, because the garbage collector automatically reclaims memory for reuse. This enables faster development with less boilerplate code, while eliminating memory leaks and other memory-related problems. At least in theory.

Ironically, Java garbage collection seems to work too well, creating and removing too many objects. Most memory-management issues are solved, but often at the cost of creating serious performance problems. Making garbage collection adaptable to all kinds of situations has led to a complex and hard-to-optimize system. To wrap your head around garbage collection, you need first to understand how memory management works in a Java Virtual Machine (JVM).

And then it continues into a lengthy treatise on JAVA's memory management. The task of sweeping out the memory garbage has been the topic of a great many academic papers, and Amazon lists two books dedicated solely to the subject:

"The Garbage Collection Handbook: The Art of Automatic Memory Management (Applied Algorithms and Data Structures)"

"Garbage Collection: Algorithms for Automatic Dynamic Memory Management"

The memory storage requirements of complex programs are extremely difficult to manage correctly by hand. A single error may lead to indeterminate and inexplicable program crashes. Worse still, failures are often unrepeatable and may surface only long after the program has been delivered to the customer. The eradication of memory errors typically consumes a substantial amount of development time. And yet the answer is relatively easy - garbage collection; removing the clutter of memory management from module interfaces, which then frees the programmer to concentrate on the problem at hand rather than low-level book-keeping details. For this reason, most modern object-oriented languages such as Smalltalk, Eiffel, Java and Dylan, are supported by garbage collection. Garbage collecting libraries are even available for such uncooperative languages as C and C++. This book considers how dynamic memory can be recycled automatically to guarantee error-free memory management. There is an abundant, but disparate, literature on the subject, largely confined to research papers. This book sets out to pool this experience in a single accessible and unified framework. Visit this book's companion Website for updates, revisions, online GC resources, bibliography and links to more GC sites.

Reviewing this book, Dr Dobb's Journal wrote: "Whatever else Java has accomplished, it has finally brought garbage collection into the mainstream. The efficiency and correctness of garbage collection algorithms is henceforth going to be of concern to hundreds of thousands of programmers; those who really care about this could do no better than to start with Garbage Collection: Algorithms for Automatic Dynamic Memory Management... the sort of comprehensive engineering manual that is so rare in computing."

The point I wanted to make, by example, rather than just waving my arms around, is that the management of memory in modern systems is actually a big deal. It's been the sole topic of many computer science PhD dissertations. Like an iceberg, most of it is hidden beneath the surface, which is a key part of what makes it "automatic" in the first place. When memory management is automatic it's designed and intended to disappear and work unseen in the background. But it turns out that pulling off that goal can be quite tricky. So it should be no surprise that one way or another we're encountering use-after-free errors in today's code.

It could certainly be that in a non-automatic language like C, one thread or process manually frees some memory that another thread or process later attempts to use. In that case, the culprit is clearly a coding error. Either the memory should not have been freed when it was, or the code attempting to still refer to it later should have known not to attempt to do so. Or, as in an instance we referred to recently, there was no actual error in the code at all. But there was a dangling pointer left around which pointed to some previously freed memory. The original code was not going to refer to that pointer, but a clever hacker figured out how to exploit it to their advantage.

What we see is that the term "Use-After-Free" is a bit of a catch-all than we would like it to be. It's a bit like when Microsoft describes a vulnerability by saying that it allowed security restrictions to be bypassed. Uhhhh... okay... anything more you'd like to say about that? No?